A Framework for Ad Hoc Interactions of Wireless Devices

Huaqun Guo+, L. H. Ngoh+, D. Q. Zhang+
+Institute for Infocomm Research, A*STAR,
21 Heng Mui Keng Terrace,
Singapore 119613
E-mail: {guohq, lhn, daqing}@i2r.a-star.edu.sg

A. K.Venkatasubramanian*, and C.Y. Wu*
*Department of Electrical & Computer Engineering,
National University of Singapore
4 Engineering Drive 3, Singapore 117576
{ajay, u0308283}@nus.edu.sg

Abstract—With the increasing pervasiveness of wireless computing technologies, users will benefit from the automatic interactions among personal devices and environments. However, current interaction systems have some limitations in terms of the problems with a centralized server, assumption of wide area wireless connectivity, or designated services instead of the universal ad hoc interaction. In this paper, we proposed a software framework for heterogeneous Bluetooth enabled devices and environments to interact with each other, based on minimum assumptions on the capabilities and infrastructure on devices & environments. The key elements of framework include Ad Hoc Sensing & Discovery Code, Mobile Code and Applications. With the framework built on Java and mobile code, we realize easily the ad hoc interactions between heterogeneous devices and environments. To demonstrate our concept, we implemented the blue_remote which uses a PDA (Personal Digital Assistant) to control a PC (Personal Computer) and alert system which allows applications in PC to make use of SMS (Short Message Service) application on PDA.

Keywords—Heterogeneous Ad Hoc Interaction, Framework, Device, Environment, Mobile Code

I. INTRODUCTION

Pervasive computing is the next generation computing environments with information & communication technology everywhere, for everyone, at all times [1]. With the increasing pervasiveness of wireless technologies, users will benefit from the interactions among heterogeneous devices and environments. For example, the elderly/disabled people use personal devices to control environment, such as opening the door, controlling the sound volume, and so on. At the same time, a PC with sensors monitors their conditions and sends SMS via mobile phone to alert doctors/relatives if they fall down or other danger happens. Hodes et al. defined Universal Interaction as allowing a device to adapt its functionality to exploit services it discovers as it moves into a new environment [2].

In this paper, we define “Device” as any personal device owned/used by a user and “Environment” as anything from a single desktop to a highly enhanced pervasive computing environment.

Ad-hoc interaction allows a mobile device to discover, interact and request available local services when it comes into a new environment, with minimum or without installation in advance. One possible scenario of the ad hoc interaction is that George, an elderly person, goes to see a specialist when he feels pain in his heart (Fig. 1). When George arrives at the hospital, his PDA interacts with the hospital environment and shows the map for him to reach the specialist room. After the specialist makes the necessary medical checks, George is advised to stay in a hospital room for observation. In the hospital room, a PC is connected to several sensors for monitoring patients and it also provides several entertainment programs for patients. When George stays in bed and is monitored by sensors in his body, he uses his PDA to select a music program in the PC and adjust the volume of sound to make him calm. At the same time, the PC collects and processes the data from sensors. If the PC detects a distress signal from George, it sends an alert message to inform doctor/nurse at once.

The above interaction scenarios are characterized by a number of challenges: devices interaction can be ad-hoc and spontaneous; devices and environments are heterogeneous; and context of user, device and environment is dynamic [3]. Our research contribution is in developing a software framework that allows heterogeneous devices and environments to interact with each other, and with minimal assumptions on the capabilities and infrastructure on devices & environments.

Figure 1. One scenario of heterogeneous ad hoc interaction.
The rest of paper is organized as follows. Section II presents the related works while Section III describes our framework for ad hoc interaction. Section IV presents the prototype of implementation and Section V describes the lessons we have learned. Finally, Section IV outlines our conclusions.

II. RELATED WORKS

There are some related works in this area: existing commercial products, emerging commercial standards and research systems for controlling appliances [4]. The Philips Pronto [5] and the Harmony remote [6] are two typical universal remote control products that can remotely control television, DVD player, and Hi-Fi set volume. The problem with commercial products is that all of the programming must be done manually, which can be a tedious and time-consuming task [4]. Microsoft-led Universal Plug and Play (UPnP) [7] and Sun’s JINI [8] are two emerging commercial standards which are targeted to reduce configuration hassles, improve device cooperation, and automatically discover the required services. However, JINI & UPnP do not provide the user interface description features described in this paper.

Among the research systems for controlling appliances, a service architecture reported in [2] was implemented to turn a client device into “a universal interactor” that adapts itself to control many devices. The key elements of architecture include: 1) augmented mobility beacons providing location information and security features, 2) an interface definition language allowing exported object interfaces to be mapped to client device control interfaces, and 3) client interfaces that maintain a layer of indication, allowing elements to be remapped as server locations change and object interactions to be composed into complex behaviors. However, there are some drawbacks in such architecture. First, a centralized server is a likely single point of failure and subject to overloading. Second, the required wireless base-station with beaconing makes it more expensive. Third, the information kept in the server may be outdated. Fourth, this mechanism does not support self-sensing or ad hoc discovery of devices.

ICrafter [9] and Xweb [10] are another two research systems for controlling appliances. ICrafter provided a framework for distributing appliance user interfaces to controller devices in the environment. Xweb investigated an infrastructure for interactive computing that was analogous to the world-wide web. However, all of those works assumed that all devices are connected to the Internet.

Among applications of Bluetooth technology for device interactions, for example, Poket Doktor project [11] was a handheld device to be used by paramedics that communicates with Bluetooth-enabled Smart Cards containing vital medical information at the scene of an accident. However, those applications of Bluetooth technology are mainly for data transfer or data synchronization. The programs have to be pre-loaded in both parties and the designated services are not suitable for the heterogeneous ad hoc interaction when a personal device comes to a new unknown environment.

In summary, the current interaction systems have some limitations in terms of the problems of single point of failure and overloading with a centralized server, assumption of wide area wireless connectivity, or designated services instead of the universal ad hoc interaction.

III. SOFTWARE FRAMEWORK FOR AD HOC INTERACTIONS

We develop a software framework for Bluetooth enabled devices and environments to interact with each other. Our minimum assumptions for ad hoc interactions are:

• communication is based on IP over Bluetooth [12, 13]
• clients have basic knowledge to understand the communication protocols with server
• Java [14] is available in devices and environments.

The framework is built on Java and uses mobile code for interaction interface. Mobile code is a software obtained from remote systems, transferred across a network, and then downloaded and executed on a local system without explicit installation or execution by the recipient [15]. There are a few major concerns regarding this approach. First of all, portability of code is a prerequisite for code mobility. Today’s devices and environments are typically heterogeneous and very often use different hardware platforms and operating systems. For these reasons, the mobile code should be portable and be able to run on each of those platforms. Maintaining a single program for many platforms is preferred to maintaining a number of programs; each for a different platform. Originally, Java was designed to implement applications that run on many platforms in heterogeneous environments [16]. This is the key reason that we have built our framework using Java technology. Secondly, security protection, which includes secure code distribution and access control lists, is also a major concern in pervasive computing environments. This is because mobile code is often active and may maliciously endanger the integrity of the managed devices.

The major components for the framework to support interactions between a PC acting as a server and a PDA acting as a client or vice versa or Peer-to-Peer interactions (PC and PDA are both servers and clients) are as follows:

• Ad Hoc Sensing & Discovery Code: A program pre-installed on the client device as well as the server which will detect nearby active Bluetooth devices, set up communication, and finally send and receive data (make the two applications talk) through the Bluetooth radio port.
• Mobile Code for simple user interface to get user’s commands/data.
• Applications which will process the commands/data received through Bluetooth, run at the Operating System (OS) level and perform the actions accordingly.

The framework involves in the field of Java Bluetooth packages called JSR 82 [17, 18] and OS level hardware control. J2ME (Java 2 Micro Edition) [19] and Swing [20] components are also used in order to develop user interfaces as mobile code. OS level hardware control in Windows has been achieved by using the simple and efficient Win32 API [21].
This API uses C and C++ natively compiled programs. Thus Java programs and C++ programs are linked through Java Native interface (JNI) [22, 23]. The framework in brief can be diagrammatically shown in Fig. 2.

In the following sections, each of the major components of the proposed framework is described.

A. Ad Hoc Sensing & Discovery Code

Ad Hoc Sensing & Discovery Code should be pre-installed on the phone/PDA as well as the PC so that clients have some basic knowledge to understand the communication protocol with server.

A Bluetooth-enabled application can be either a server or a client – a producer of services or a consumer – or it can behave as a true peer-to-peer endpoint by exposing both server and client behavior. All Bluetooth-enabled applications must first initialize the Bluetooth stack. A server further makes services available to clients. It then waits for incoming connections, accepts them as they come in, and serves the clients. A client first discovers any nearby devices, then for each discovered device it searches for services of interest.

This Ad Hoc Sensing & Discovery Code is the same for all interactions. All the interactions share this program and call functions from here to establish and communicate through the Bluetooth channel. The programs on the PDA access this directly through packaging since its applications have been built on J2ME.

B. Mobile Code

Mobile code allows easy and efficient interactions between heterogeneous personal devices (e.g., PDA) and pervasion computing environments (e.g., PCs) [3]. The whole interaction sequence as shown in Fig. 3 is described next. An Ad Hoc Sensing & Discovery Code will be pre-installed at all client devices and all servers so that a client device and a server can have some basic knowledge for the communication with one another. The server will periodically broadcast a Hello message. After the Hello message appears in the client side, the user will respond to the hello message to the server to indicate his intention for further interactions. The server will send the mobile code which implements a user interface to the client. The mobile code will then run in the client device and the user interface will appear in the client device to allow the user to interact with the server. The user uses the keypad or stylus to click on the user interface to issue commands which will be captured by the Ad Hoc Sensing & Discovery Code as string or Boolean types (depending on the requirement of the action to be performed). Finally, when the server receives the commands, its corresponding application will take the action to meet the user’s needs.

The advantages with mobile code are:
- Flexible – the server provides the mobile code to the client side without explicit installation or execution by the client (instead of having the user to spend efforts to install different software programs for interaction with different devices/environments)

C. Applications

These hardware applications need to work at the OS level since it has to interact with java applications. The OS used here is Windows XP professional SP2. These programs which make use of the Win32 API are developed in C++. These C++ applications interoperate with the Java Bluetooth application using JNI. JNI is a programming framework that allows Java code running in the Java virtual machine (VM) to call and be called by native applications (programs specific to a hardware and operating system platform) and libraries written in other languages, such as C, C++ and assembly.

Putting it clearly, the Java Ad Hoc Sensing & Discovery Code uses the JNI framework to access the Win32 API through C++ codes. The command that it receives from the client through Bluetooth is processed and the appropriate action is performed through the C++ dynamic link libraries (DLLs) file.

IV. IMPLEMENTATIONS

We have developed a prototype based on the software framework described above in order for Bluetooth enabled PDAs and PCs to interact with each other mainly to blue_remote and alert system.

A. Blue_Remote

The blue_remote is simply a solution to use PDAs for controlling PC through Bluetooth. Controlling mouse with touch screen stylus movement, setting volume of PC, and locking/shutting down the PC with a single click are a few
among the many other possible applications. This will help the PDA users to control their PCs when they are away from their computers and within the range of 10 meters which is the range for Bluetooth Class 2 Radios [12]. Here are the brief descriptions of each of the demo implementations.

- **Virtual Mouse Pad** - PDA is used as a virtual mouse pad, which means a user can use the stylus of his PDA to point, drag and click on the PDA touch screen to remotely control the motion of the mouse cursor on the PC screen.
- **Volume control application** - PDA is used as a remote volume controller to adjust the volume level and left-right balance of the sound coming out from the speakers connected to the PC.
- **PC Shutdown** - PDA is used as a remote switch to shut down PC by just one touch on the soft button on PDA touch screen.

The virtual mouse pad has a canvas on which the touch screen stylus can be scrolled (Fig. 4). Quite clearly from the UI (User Interface) scrolling on the canvas will move the mouse on the PC screen. Each mouse move event is trapped and the parameters acquired are sent to the PC. The change in mouse position is simply reflected on the PC side. The buttons below the scrolling canvas perform single click, double click and right click respectively. For example, a user can click on a music program through virtual mouse pad in his PDA.

The volume control application has 4 buttons to increase the volume and adjust the balance (Fig. 5). Up and down buttons increase and decrease the volume. The left and right buttons set the balance. Similar to the previous application, clicking these buttons are captured and the parameters are sent to the PC application.

The interface confirms with user for shut down PC (Fig. 6). Clicking on Yes shows a progress bar that updates the completion of the task. The PC is then shutdown. Lock screen can also be implemented under the same application.

### B. Alert System

Alert system allows applications in PC to make use of SMS application on the PDA. It is a Bluetooth based messaging system which provides the platform for developing intelligent alert systems where PC sends out alert or warning signals to mobile device via Bluetooth. This application can be useful in the hospital environment where doctors can receive alerts on their mobile phone after the server gathering and processing the information from patients. The Bluetooth communication part is similar to what we have done for blue_remote except that the server and client here are reversed. However, different from other applications we implemented before, the alert system application must be always active at the background, no matter what application is currently running at the foreground. Whenever any warning message arrives via Bluetooth, it should be ready to accept immediately and the pop-up message will immediately interrupt whatever current application which is running (Fig. 7).

After PC acquires sensor information and processes the acquired data, PC uses the alert system to send the message to the PDA through Bluetooth and wrap it as an SMS for the sake of convenience. The received alert message can also be stored with other messages in the SMS inbox.
The alert system will save a lot of work loads for nurses since nurses usually must collect the data manually and monitor the condition of patients from time to time.

V. LESSONS LEARNED

There are some lessons that we have learned during the prototype implementation.

First, Bluetooth signals are omni-directional and can pass through walls and briefcases. Communicating devices do not need to be aligned and do not need an unobstructed line of sight. On the other hand, the ad hoc interaction enables, for example, the elderly/disabled people to use mobile phones/PDAs to control the environment equipments. Thus, Bluetooth is good technology for ad hoc interaction.

Second, during the implementation of alert system, a J2ME alert and trigger work well only when the PDA is in active mode because the operating system of the PDA only supports a few active system applications by default. Thus, it is better to use Wireless Message API of J2ME, where we wrapped the alert message received from Bluetooth and sent as a SMS to the PDA itself at the background. By doing it, mobile user can receive the alert message just like receiving normal SMS.

Finally, to prevent unauthorized access, the interaction should consider a sufficient level of security, privacy and trust to meet the user's needs. The interactions and information exchange must be secure, private and trustworthy. This is the area that we are currently working on.

VI. CONCLUSIONS

To overcome current interaction systems’ limitations in terms of the problems with a centralized server, assumption of wide area wireless connectivity, or designated services instead of the universal ad hoc interaction, this paper proposed a software framework for heterogeneous ad hoc interactions. The novelty of the framework is that with the minimum capability requirements on devices & environment, users are able to interact easily between their personal devices and environments and benefit from their interactions.

The software framework is built on Java and mobile code which is portable and is able to run on different platforms. The key elements of framework are to design Ad Hoc Sensing & Discovery Code, Mobile Code and Applications. In our prototype, we develop blue_remote and alert system to demonstrate the feasibility of our framework. The blue_remote is to use PDA for controlling PC through Bluetooth. Alert system allows applications in PC to make use of SMS application on PDA.

For our future work, we will work on security, privacy and trust of ad hoc interaction.

REFERENCES